«Санкт-Петербургский государственный университет»

Математическое обеспечение и администрирование информационных систем

Кафедра информационно-аналитических систем

Шпарута Софья Константиновна

Методы автоматизированной генерации программного кода по тексту на естественном языке

Бакалаврская работа

Научный руководитель:

канд. физ.-мат. наук

доцент кафедры ИАС

Графеева Наталья Генриховна

Рецензент:

ведущий инженер Отдела автоматизации PVG

Романова Светлана Александровна

Санкт-Петербург

2018

SAINT-PETERSBURG STATE UNIVERSITY

Mathematical support and administration of information systems

Sub-Department of Analytical Information Systems

Sofia Shparuta

Methods of automated generation of program code for text in natural language

Bachelor's Thesis

Scientific supervisor:

P.H.D.

associate Professor of sub-department of AIS

Natalia Grafeyeva

Reviewer:

leading engineer of Automation department of PVG

Romanova Svetlana

Saint-Petersburg

2018

Оглавление

Введение

В наши дни очень актуальна проблема того, что люди, не обладающие какими-либо знаниями информационных технологий, не могут воспользоваться инновациями. Даже ввести что-то в строку Google, например, для людей пожилого возраста – большая проблема. Сейчас существует очень много решений данной проблемы. Но, в основном это реализовано для каких-то глобальных систем. А системы меньшего масштаба все еще требуют каких-либо знаний языков запросов, программирования и просто технических знаний. А вот как пользоваться такой системой “непросвещенным” людям?

Постановка задачи

Из сказанного во введении ясно, что разработка системы, которая позволит выдавать ответы на вопросы на естественном языке, но при этом будет небольших масштабов, может помочь людям без технических знаний работать с базой данных. Также ее будет несложно внедрять в другие программы, благодаря небольшим размерам.

Отсюда вытекает задача данной работы:

На основе существующей базы данных написать приложение, которое получает на вход вопрос на естественном языке, переводит его в SQL-запрос и выдает пользователю его результат.

Исходные данные

Из скриптов, предоставленных студентами третьего курса кафедры ИАС математико-механического факультета, получаем базу данных MIGRATION_EXPERT, которая содержат информацию об экспертах, работающих в миграционных службах, и их публикациях соответственно (см. Рис. 1).

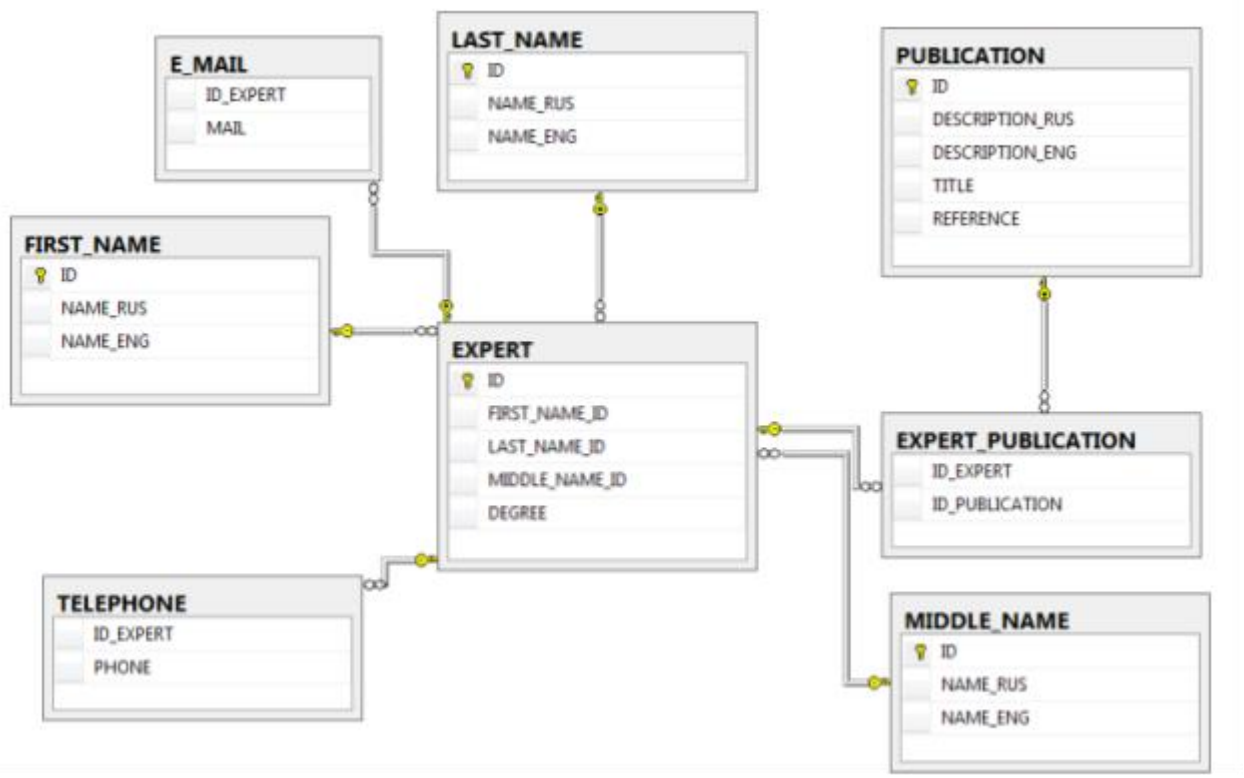

Рис.1: Диаграмма базы данных MIGRATION_EXPERT

Обзор существующих решений

Проблему, сформулированную в предыдущем разделе, изучало большое количество авторов. Рассмотрим небольшую выборку из их статей.

Dataset for a Neural Natural Language Interface for Databases ***[2]***

В данной статье авторы предоставили новый тренировочный датасет SENLIDB (Stack Exchange Natural Language Interface to Database) для разработки модели машинного обучения, основанной на данных.

В статье есть таблица (см. Рис. 2), где приведены несколько примеров SQL-запросов, сгенерированных для тестового набора. Во втором примере мы можем заметить, что модель учится правильно использовать псевдонимы таблиц в сложных запросах с объединенными таблицами. С другой стороны, хотя сгенерированные запросы являются синтаксически правильными, в большинстве случаев они не могут вернуть желаемые результаты при их выполнении.

Natural language description	Predicted SQL query
top 50 users from india	select top 50 id , displayname , reputation , web-siteurl , location from users where location like india order by reputation desc
top 100 users with no of gold badge	select top 100 b . userid , u . displayname , count (*) as gold from badges b inner join users u on u . id = b . userid group by b . userid , u . displayname order by count (b . name) desc
user rep	select reputation , count (*) from users group by reputation order by reputation
text of highest - scored comment	select top 100 id as [comment link] , score , text from comments order by score desc
questions with most view	select top 500 id as [post link] , viewcount from posts order by viewcount desc

Рис. 2: Запросы на естественном языке и сопоставленные им SQL-запросы

Авторы решили оценить способность предлагаемого NNLIDB правильно создавать таблицы и столбцы из схемы базы данных. Наблюдается, что при проверке показатели точности и полноты значительно достаточно высоки.

Interactive Natural Language Query Construction for Report Generation ***[3]***

Здесь авторам нужно понять, какие языковые знания необходимы для правильной интерпретации вопросов пользователей. Утонченность

пользователя и среды, в которой используется система, также повлияет на то, как пользователи будут вводить данные в систему. Наконец, нужно решить, как наилучшим образом отвечать на вопросы пользователя, будь то выборка существующих документов, динамическое создание отчетов с структурированной базой данных или создание предложений на естественном языке.

Позиция, которая принимается авторами, сильно влияет на дизайнерские решения, простоту конфигурации и опыт конечных пользователей. Однако, можно рассмотреть анализ естественного языка как задачу, связанную с генерацией естественного языка, где пользователи создают вопросы естественного языка, делая выбор, который добавляет слова и фразы. Данным образом пользователи строят запросы в управляемом меню, чтобы задавать вопросы, которые всегда однозначны и понятны для любого человека, получая ответы в виде интерактивных отчетов базы данных (а не текстовые отчеты). Используя технику на основе меню, пользователь вынужден вводить только действительные и хорошо сформированные запросы. Сложность системы значительно сокращается, поскольку язык, который система должна обрабатывать, прост и однозначен.

Цель авторов состояла в том, чтобы создать грамматику, подходящую для приложения для запросов к базе данных. Семантическая модель использует и сущности (унарные предикаты), и отношения (бинарные предикаты), которые могут быть непосредственно и автоматически сопоставлены с лексическими элементами и фразами, которые пользователь видит на экране во время построения запроса. Когда пользователь завершил построение запроса на естественном языке, создается соответствующая логическая формула первого порядка, которая затем может быть переведена в запрос базы данных.

Предположение авторов заключалось в том, что многие запросы к базе данных могут быть выражены в определенном шаблоне,

Для оценки такой системы требуется изучение трех показателей эффективности: охват доменов, простота использования и эффективность запросов.

В первоначальной оценке авторы применили эти показатели к выбору 12 вопросов естественного языка к данным в базе данных Adventure Works, на которые можно было бы ответить с помощью системы построения запросов на естественном языке. Эти вопросы были сгенерированы пользователем с предварительным воздействием на базу данных Adventure Works, но не были предварительно подвергнуты поисковой системе программного обеспечения или ее дизайну, или алгоритмам, поэтому вопросы не намеренно настраиваются для получения искусственно оптимальных результатов. Восемь из этих вопросов были непосредственно подотчетны, а четыре были косвенно подотчетны. По каждому из этих вопросов авторы измеряли количество кликов, необходимых для генерации строки запроса, время, необходимое для выполнения требуемых кликов, и время, необходимое для получения необходимых записей и создания отчета. Различие между непосредственно подотчетным и косвенно подотчетным вопросами заслуживает краткого объяснения. Вопрос считается *непосредственно подотчетным*, если ответ является единственным результатом, возвращенным в отчете, или если ответ включен в группу результатов. Вопрос считается *косвенно подотчетным*, если отчет, созданный на основе соответствующего запроса, может использоваться для вычисления ответа или если информация, относящаяся к ответу, является подмножеством возвращаемой информации.

Таким образом, оценка показывает, что подход, основанный на меню NLG, приводит к быстрому созданию однозначных запросов, которые могут получить соответствующую информацию базы данных, соответствующую запросу. Этот подход имеет ограничения, поскольку у пользователей, вероятно, будут вопросы, которые либо не могут быть построены семантической грамматикой, либо не могут быть получены из базовой базы

данных. Тем не менее, проблемы, связанные с выбором или двусмысленностью, которые часто встречаются системами NLG в частности, и системами обработки естественного языка в целом, можно избежать. Отчеты базы данных генерируются быстро, обеспечивая однозначные ответы в ясном, гибком формате и в знакомой, удобной, незапугивающей веб-среде. Помимо преимуществ удобства использования, этот веб-подход имеет дополнительное преимущество для минимизации конфигурации и обслуживания. Результаты только наводят на размышления, поскольку они включают только 12 вопросов. Считается целесообразным расходовать ресурсы для полного исследования, которое включает в себя несколько пользователей с различным уровнем опыта, несколькими доменами и более широкими наборами вопросов.

Natural Language Query System for RDF Repositories ***[1]***

Основная цель данной работы заключалась в изучении путей улучшения переносимости интерфейсов естественного языка в разных базах данных. Авторы разработали и внедрили систему Natural Query, которая может эффективно ранжировать различные семантические интерпретации вопросов естественного языка. Начали авторы с анализа проблемы архитектуры программного обеспечения, которая возникает в результате необходимости обработки большого объема информации. Необходимо обеспечить разделение функциональности и информации таким образом, чтобы значительная часть сложности лингвистической обработки могла быть полностью разделена и не зависела от конкретного домена, организации и содержимого хранилища данных.

Система NQ обрабатывает вопрос естественного языка в пять этапов.

1. Семантическая маркировка

Этап семантической маркировки выполняет две различные функции: маркировка значений и маркировка категорий. Маркировка значений помещает все многоточечные токены в вопросе, которые соответствуют значениям, хранящимся в базе данных. Данный процесс также включает в себя

распознавание выражений для регулярных упорядоченных типов значений, таких как числа, время и дату. В некоторых случаях значения в вопросе используются для формирования вопросов включения интервалов, например, «кто послал мне приглашение на ужин на прошлой неделе?» В этом вопросе «на прошлой неделе» переводится в интервал данных, концы которого могут или не могут присутствовать как явные значения в репозитории. Синтаксически оба типа тегов значений выглядят одинаково. Маркировка категории в свою очередь идентифицирует потенциальные ссылки на объекты базы данных, такие как классы и свойства

2. Разбор

Здесь решается некоторая двусмысленность, которая может появиться на предыдущем этапе. В этом процессе парсер идентифицирует фокус вопроса и наиболее вероятную привязанность фраз. Использование общей грамматики может быть проблематичным в некоторых областях, и планируется исследовать генерацию семантической грамматики с использованием организации базы данных, языковых тегов и некоторых дополнительных аннотаций грамматики, связанных с объектами базы данных.

3. Абстрактная семантическая интерпретация

Для каждого разбора вопроса мы создаем абстрактную семантическую интерпретацию или представление смысла. Благодаря семантическим методам, выполняемым парсером, например, для вопроса “What are the names of my contacts at IBM in Ulm?”, мы теперь знаем, что

1. «Ulm» – значение категории Locality.name

2. «IBM» – это значение категории Organization.name

3. “contact” – это ссылка на класс Contact

4. “name” – это ссылка на одно из многих атрибутов имени, которое является синтаксическим анализатором для контакта. Эта информация достаточна для создания абстрактного представления, показанного ниже, с использованием предикатов бинарных префиксов, которые соответствуют категориям баз данных:

(x name ?v)

(x type Contact)

(x Organization.name IBM)

(x Locality.name Ulm)

В системе NQ интерпретация представленного выше представления означает «вернуть значения атрибутов, помеченных как «имя» экземпляра класса, помеченного как «Контакт», связанного с помощью свойств, помеченных как «Organization.name» и «Locality.name», к значениям «IBM» и «Ulm» соответственно».

4. Конкретная семантическая интерпретация

В приведенном выше примере представление значения содержит классы Contact и Organization, а также значение «Ulm» свойства name экземпляра Locality и «IBM» в качестве значения свойства name экземпляра класса Organization.

5. Эвристическое ранжирование

Наиболее привлекательно основать ранжирование на структурных свойствах полученных подграфов. Это сделало бы систему ранжирования независимой и переносимой в разных хранилищах. Но это возможно только в том случае, если есть ограничения на структуру самого графа базы данных. Другими словами, можно использовать структуру подграфов базы данных для семантического ранжирования, если структура в первую очередь имеет семантическое значение.

Самый обширный эксперимент был в области поиска работы. В репозитории содержится подробное описание более 70 000 предложений о работе. Авторы собрали более 5000 различных вопросов естественного языка для домена. Все вопросы сопоставлены с 300 различными запросами. Все запросы выполняются успешно. Мы проверили результаты из 25 запросов, соответствующих примерно 200 вопросам. Система продемонстрировала стопроцентную точность и почти стопроцентную полноту.

Question Answering System in a Business Intelligence Context ***[4]***

В данной статье речь идет о модели предсказания. Первым шагом идет кластеризация запросов. Поскольку размер истории в динамической среде может быть довольно большим, кластеризация похожих запросов для пользователя представляет собой подходящее решение для уменьшения размера данных и перехода к допустимому числу состояний, которые не являются слишком общими и не слишком конкретными. Кроме того, анализ кластеров позволяет лучше определить привычки пользователей, которые могут улучшить точность нашего механизма прогнозирования. На входе этого шага подаются векторы, представляющие запросы пользователей, хранящиеся в истории. Основной задачей кластеризации является обнаружение повторяющихся и похожих запросов из всех запросов в истории.

Далее, анализируя историю, представленную триплетом < намерение; контекст; предпочтения > механизм прогнозирования может изучить модель поведения пользователей в динамической среде и, следовательно, вывести наиболее вероятный следующий запрос. Выводится наиболее вероятный следующий запрос, моделируя поведение пользовательского запроса как цепи Маркова. Каждый кластер запросов представляет собой состояние в связанной цепи Маркова. Ряд состояний системы имеет свойство Маркова. Поэтому прошлые состояния не дают никакой информации о будущих состояниях. Если машина находится в состоянии x в момент времени n, вероятность того, что она переместится в состояние y в момент n + 1, зависит только от текущего состояния x, а не от прошлых состояний. Распределение вероятности перехода можно представить в виде матрицы P, называемой матрицей перехода, причем (i, j) - й элемент из P равен:

$$P_{ij} = \Pr(X_{n+1} = j \mid X_n = i)$$

Начальная вероятность $\Pr(X_{n+1} = j \mid X_n = i)$ равна $\frac{1}{m}$, где m - количество запросов, которые могут следовать текущему запросу Q_i. $\Pr(X_{n+1} = j \mid X_n = i)$ можно было бы обновить, подсчитав, как часто запрос Q_j предшествует

запросу Q_i и делит это число на общее количество запросов, которые наблюдались как следующий запрос Q_i. Это означает, однако, что прошлое так же важно, как и настоящее.

В большинстве случаев серия запросов, которые выполняет пользователь, будет развиваться во времени. Например, если история рассматривает запросы, выполненные пользователем в течение последних шести месяцев, то мы хотим, чтобы более свежие запросы имели относительно большее влияние на модель поведения пользователя, чем более старые запросы. Следовательно, функция вероятности перехода должна обновляться таким образом, чтобы последние переходы имели большее значение, чем более старые. Для этого метод экспоненциального сглаживания можно использовать так, чтобы прошлое взвешивалось с экспоненциально уменьшающимися весами. Это может потребоваться в быстро меняющейся среде или, когда система развернута и начинает учиться. В довольно статической среде могут быть установлены на низком уровне. Итак, включив скорость обучения, можно убедиться, что модель поведения пользователя динамична и развивается вместе с меняющимися привычками или предпочтениями пользователя.

В следующей главе описываются эксперименты и их результаты. Рассматривается конструктор запросов, который просто представляет текстовое окно поиска пользователю. По мере того, как пользователь набирает, ему предлагаются автозаполнения. На Рис. 3 показаны, названия

Рис. 3: Интерфейс системы

объектов, которые предлагаются, когда пользователь начинает вводить «sa»: ‘Sales revenue’, ‘Avg of savegoal’ and ‘Keeper save goal’. На правой стороне

рисунка пользователь выбрал первое предложение (т. е. ‘Sales revenue’) и продолжает печатать «с». Система предлагает два измерения: ‘City’ and ‘Category’. Для инициализации автоматического завершения требуется, чтобы пользователь точно знал имена объектов, которыми он хочет манипулировать, что может стать препятствием для принятия. Чтобы помочь ему начать работу и изучить доступные данные, предложения могут быть всплывающими для пользователя, прежде чем он даже начнет печатать. Например, можно было бы предложить наиболее часто используемые меры и размеры различных моделей доменов. На Рис. 4 представлены пять атрибутов, которые наиболее часто встречаются с определенной мерой с названием “Sales Revenue”.

Предложенная система представлена в виде интерфейса поиска, доступного как на компьютере, так и на мобильных устройствах.

Measure	Dimension	Co-occurrence
Sales revenue	Quarter	0,38
	State	0,25
	Year	0,25
	Category	0,25
	Lines	0,22

Рис. 4: Атрибуты, которые наиболее часто встречаются с определенной мерой

Пользователи вводят или произносят запросы в своих собственных условиях, и в конечном итоге результаты поиска отображаются в приборной панели. Авторы сосредоточились на этом тезисе о переводе вопросов в NL в запросах базы данных.

Методология работы

По итогам обзора было решено, что решение, используемое в работе **[3]** наиболее подходящее, и, согласно этому подходу, трансляция вопроса на естественном английском языке в SQL-запрос будет производиться в 4 этапа, аналогичным первым четырем этапам в работе **[3]**.

До появления вопроса на естественном языке проводится препроцессинг. После его завершения, выполняется тегирование, парсинг, абстрактная семантическая интерпретация, конкретная семантическая интерпретация.

Препроцессинг - предварительная обработка базы

Основное назначение препроцессинг связать слова естественного языка и формального. Назовем *служебными словами* артикли, например, a, an, the, вопросительные слова, например, What, Who, Which. Будем называть *словами формального языка* слова, соответствующие названиям объектов базы данных, например, название таблицы или колонки. Также нужно понять, какое слово в вопросе какую роль играет (Абстрактная семантическая интерпретация). Для этого на данном этапе будет строиться грамматика. Иными словами, инициализируем *лексикон*, т.е. набор слов, известных системе, и метки, которые нужны для перевода запроса из естественного языка в формальный.

Сначала запишем синонимы к названиям таблиц и колонок, которые есть в нашей в базе данных. Например, таблице PUBLICATION будут сопоставлены слова из естественного языка “publication” и “work”, колонке PUBLICATION.TITLE – “title”.

Далее при помощи запросов вида *select * from table* ко всем таблицам базы, получим значения, хранящиеся в базе данных. Полученный кортеж разбивается на отдельные значения (будем в дальнейшем называть *словами*) и в виде структуры типа <слово, таблица.колонка> сохраняется в лексикон.

Например, для запроса select * from expert получаем:

	ID	FIRST_NAME_ID	LAST_NAME_ID	MIDDLE_NAME_ID	DEGREE
1	2	7	1	7	Кандидат наук
2	3	11	7	11	Доктор наук
3	4	13	6	2	Магистр
4	5	5	3	4	Кандидат наук

<Кандидат наук, Expert.Degree>, <2, Expert.ID> и т.д.

Затем, при помощи запросов к базе, получим информацию о внешних ключах в базе и сохраним её в виде графа. В дальнейшем эта информация понадобится для последнего этапа – конкретной семантической интерпретации.

Переходим к следующему шагу препроцессинга. Построим грамматику, взяв за основу правила, используемые в **[5]**. Она должна позволять анализировать определенные типы вопросов, предусмотренные системой (см. Приложение 1). Опишем ее формально:

G = <∑, N, P, S>

∑ – Набор терминальных символов:

P - имя существительное собственное

A - прилагательное

N - имя существительное

I - глагол непереходный

BEs - is, was

BEp - are, were

AR - a, an, the

AND - and

WHO - Who

WHOs - who

wHICH - which

WHAT – What, Which

OR - or

Of – of, by

LIKE - like, about

With - with,

‘WHOSE’ - Whose

HAVE - has, have

N - Набор нетерминальных символов:

{VP, AN, Rel, OF, BE, NP, Nom, WITH, WHOSE}

P - Набор правил:

VP -> T NP|BE A|VP AND VP|LIKE NP|BE NP|NP BE I",

NP -> P|AR Nom|Nom",

Nom -> AN|AN Rel",

AN -> N|A AN",

Rel -> WHO VP|NP T",

BE -> BEs|BEp",

OF -> WHO HAVE",

WITH -> WHICH HAVE",

WHOSE -> WHOs HAVE|WHOSE I",

S -> WHAT VP OF NP WITH NP VP|

WHAT VP OF NP OF NP WITH NP VP|

WHAT VP OF NP OF NP VP|WHAT VP OF NP|

WHAT QP OF NP|WHAT VP OF NP VP|WHOSE NP VP|

WHAT VP THERE|WHOSE NP BE VP|

WHOSE NP WITH NP VP"

S - Начальный нетерминал.

На этом предварительная подготовка закончилась.

Этап 1: Тегирование

Данный этап состоит в том, чтобы сопоставить словам, которые не являются служебными в вопросе на естественном языке, метки, которые представляют собой пару: значение и категория.

a) Значение – слово, из лексикона.
b) Категория – показывает, является ли слово значением данного объекта базы (указанной в метке-значение), или оно является названием этого самого объекта, т.е. она принимает 2 значения: {название объекта, значение}.

Поясним эти понятия на примере. Есть вопрос на естественном языке: *What is the **e-mail** of **Ann**?* Жирным курсивом выделены слова, не являющиеся служебными в данном вопросе. Они будут разобраны следующим образом (см. Таблицу 1).

Таблица 1: Метки, соответствующие неслужебным словам

Слово	Метка-значение	Метка-категория
e-mail	E_MAIL	название объекта
Ann	FIRST_NAME.NAME_ENG	значение

В этом вопросе слово e-mail имеет метку (название объекта, “E_MAIL”), которая говорит о том, что это название таблицы E_MAIL, а слово Ann имеет метку (значение, “FIRST_NAME.NAME_ENG”), которая говорит, что это слово является значением поля NAME_ENG таблицы FIRST_NAME.

Этап 2: Парсинг

Парсинг позволяет понять так называемый фокус вопроса, т.е. что в вопросе является условием поиска, а что предметом поиска.

Каждому слову в вопросе сопоставляем терминальные символы. из грамматики, созданной в препроцессинге. Затем, по правилам грамматики, указанной в препроцессинге, строится дерево парсинга.

Например, для вопроса “What is the e-mail of Ann?” будет построено ниже указанное дерево (см. Рис. 5).

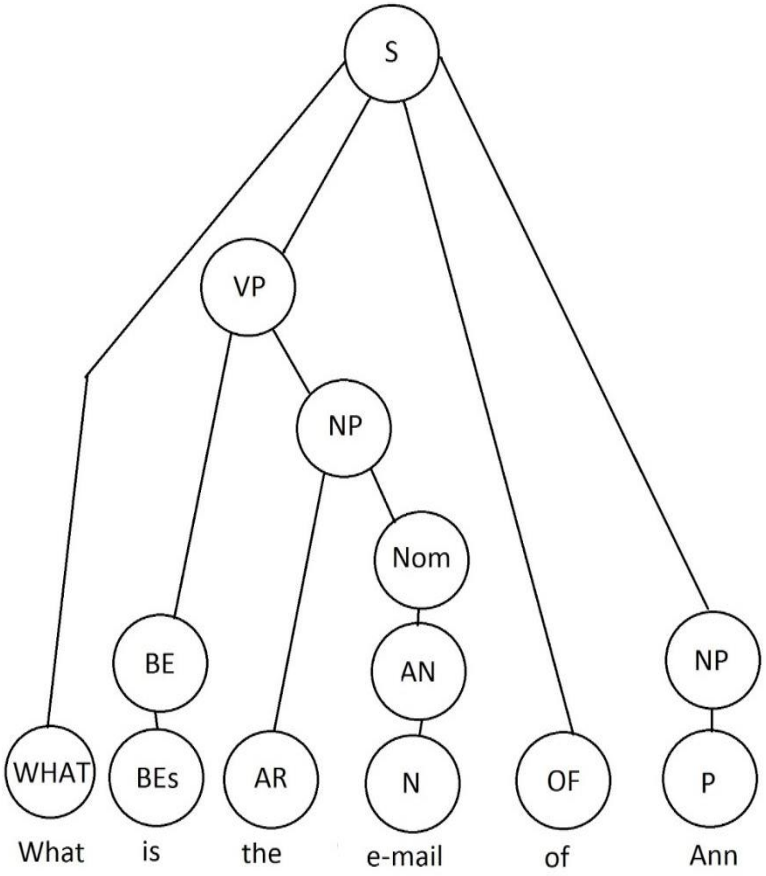

Рис. 5: Дерево парсинга для вопроса «What is the e-mail of Ann?»

Этап 3: Абстрактная семантическая интерпретация

Основное назначение данного этапа – сформировать абстрактные фрагменты последующего SQL-запроса, где будет явно отражено, что ищется, где ищется (таблица), при каких условиях (в каких таблицах). Из полученного дерева парсинга, например, для вопроса “What is the e-mail of Ann?”, мы должны узнать, что ищется информация о e-mail (e-mail ?v), которая содержится в таблице E_MAIL (type E_MAIL). И эта информация должна быть связана с именем Ann - значением поля Name_eng таблицы First_name (First_name.Name_eng Ann). Этого достаточно для создания абстрактного представления, показанного в Таблице 4.

Таблица 2: Абстрактные фрагменты и их назначения

Абстрактный фрагмент	Что указывает
e-mail ?v	Что
type E_MAIL	Где
First_name.Name_eng Ann	Какие условия и в каких таблицах

Этап 4: Конкретная семантическая интерпретация

Из полученной на предыдущем этапе информации, можем составить фрагменты запроса: *select*, *from* и *where* (см. Таблицу 3).

Таблица 3: Фрагменты запроса и соответствующие им предложения SQL

Фрагмент запроса	Предложение
mail	select
E_MAIL	from
First_name.Name_eng = Ann	where

Но этого еще недостаточно для построения запроса. Далее нужно понять, как составить соединение таблиц и нужно ли оно вообще. С помощью графа базы данных, упомянутого в препроцессинге, можно уловить связь таблиц (наличие внешнего ключа), если это нужно. А нужно это или нет, также помогает понять информация из предыдущего этапа. Там также указаны используемые при поиске таблицы.

Для данного примера, это таблицы E_MAIL, используемая в конструкции *from*, и FIRST_NAME, используемая в конструкции *where*. Это разные таблицы, а значит, нужно добавить для них условие соединения. В данном случае условие соединения выглядит так: *FIRST_NAME join EXPERT on FIRST_NAME.ID = EXPERT.FIRST_NAME_ID join E_MAIL on EXPERT.ID = E_MAIL.ID_EXPERT*

Итоговый запрос: *select MAIL from FIRST_NAME join EXPERT on FIRST_NAME.ID = EXPERT.FIRST_NAME_ID join E_MAIL on EXPERT.ID = E_MAIL.ID_EXPERT where FIRST_NAME.NAME_ENG like '%Ann%'*

Реализация

Данная работа реализована в виде пользовательского приложения для Windows на языке C# с использованием функций из работы **[5]**. Приложения содержит в себе 3 проекта: UI, библиотека трансляции, и тестовый проект. Библиотека трансляции представляет собой dll-библиотеку на C#.

Для построения лексикона заведен класс dbGraph, котором хранится информация о БД, таблицах и связях между ними. Также он помогает уловить связи таблиц на этапе 4 (см. пред. раздел). В конструкторе прописано построение самого графа базы данных. На Рис. 6 ниже можно увидеть строение узла графа – класс Table.

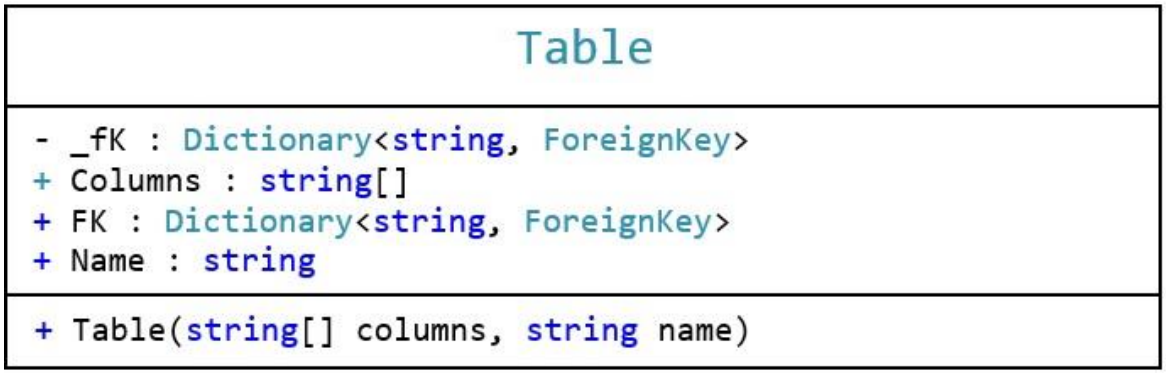

Рис. 6: Описание класса Table

С помощью экземпляра класса *SqlConnection* происходит подключение к БД и выгрузка имен таблиц, названий их колонок и информации о внешних ключах. Затем вся эта информация заносится в граф – массив экземпляров класса *Table*. Также класс *dbGraph* содержит метод *ReturnDataValues*, который по названию таблицы возвращает коллекцию меток в формате (значение, “таблица.колонка”). В данном приложении заведен класс *Lexicon*. Описание на Рис. 7.

В этом классе хранится коллекция слов, помеченная тремя видами тэгов: тег-группа, тег-метка и POS-тэг. Тэг-группа имеет 2 вида: Значение(Value) и Объект (Object). Языковые тэги (language-tags) загружаются также с помощью dbGraph. Таким образом реализован 1 этап.

Для грамматики заведен класс Grammar (см. Рис. 8), который содержит в себе правила и методы, для построения дерева парсинга.

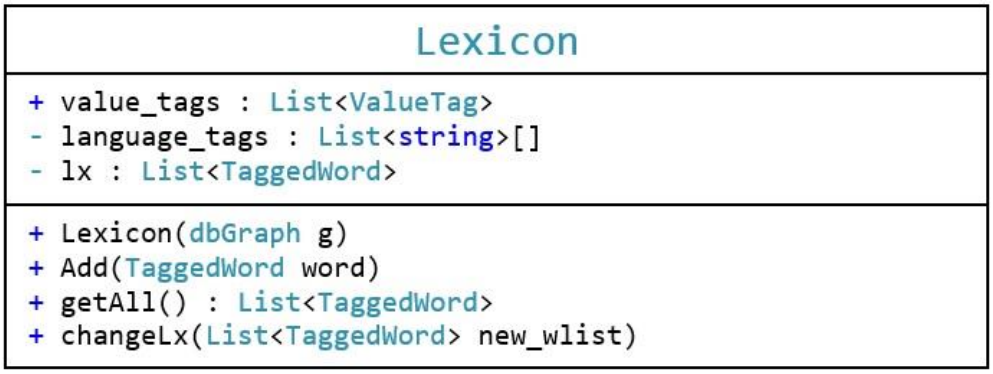

Рис.7: Описание класса Lexicon

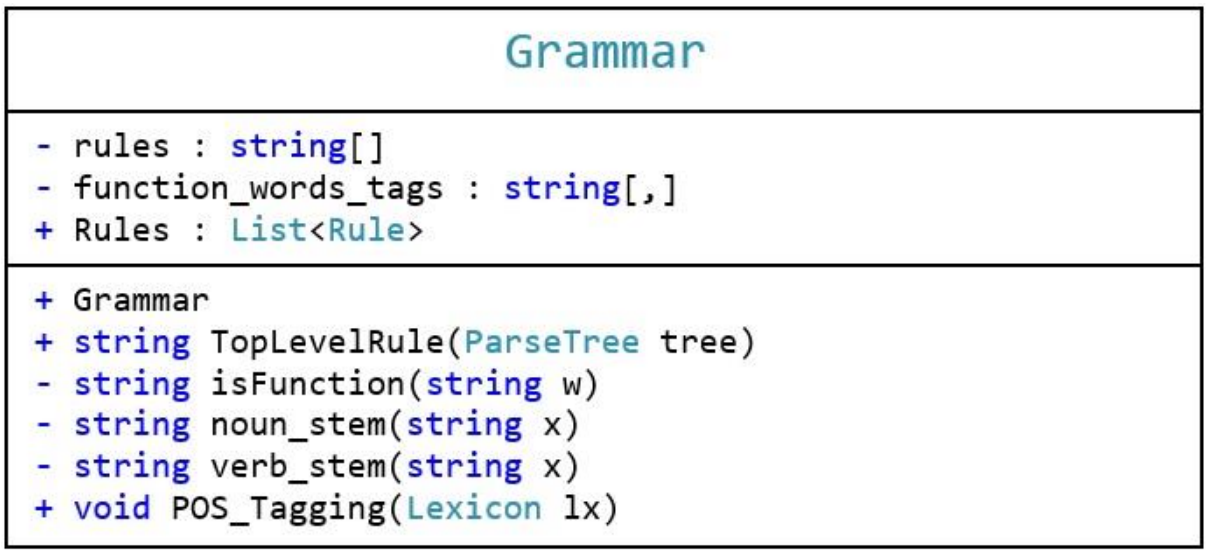

Рис.8: Описание класса Grammar

Сначала, каждому слову приписываются, так называемые POS-тэги в методе *POS_Tagging*, которые показывают принадлежность определенной части речи. После чего из них строится узел *ParseNode*. В итоге получается, так называемое, дерево парсинга, которое представлено классом *ParseTree*

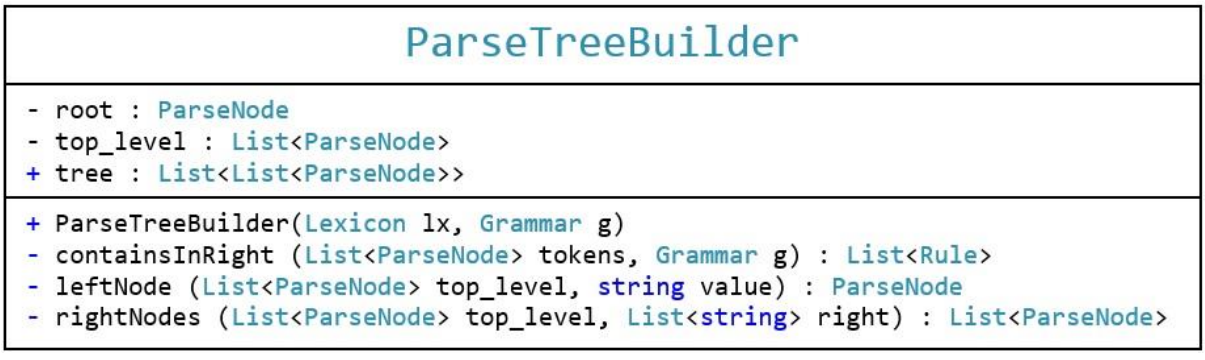

Рис.9: Описание класса ParseTreeBuilder

(см. Рис. 10), которое для построения непосредственно дерева использует *ParseTreeBuilder* (см. Рис. 9).

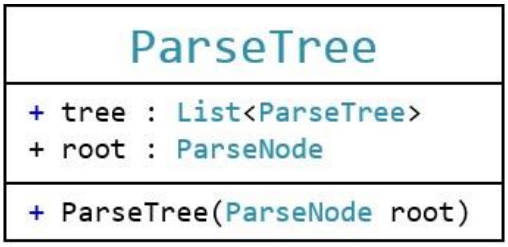

Рис.10: Описание класса Parse Tree

Все этапы из предыдущего раздела собраны в одном классе Translation, который на вход получает вопрос и поэтапно обрабатывает его.

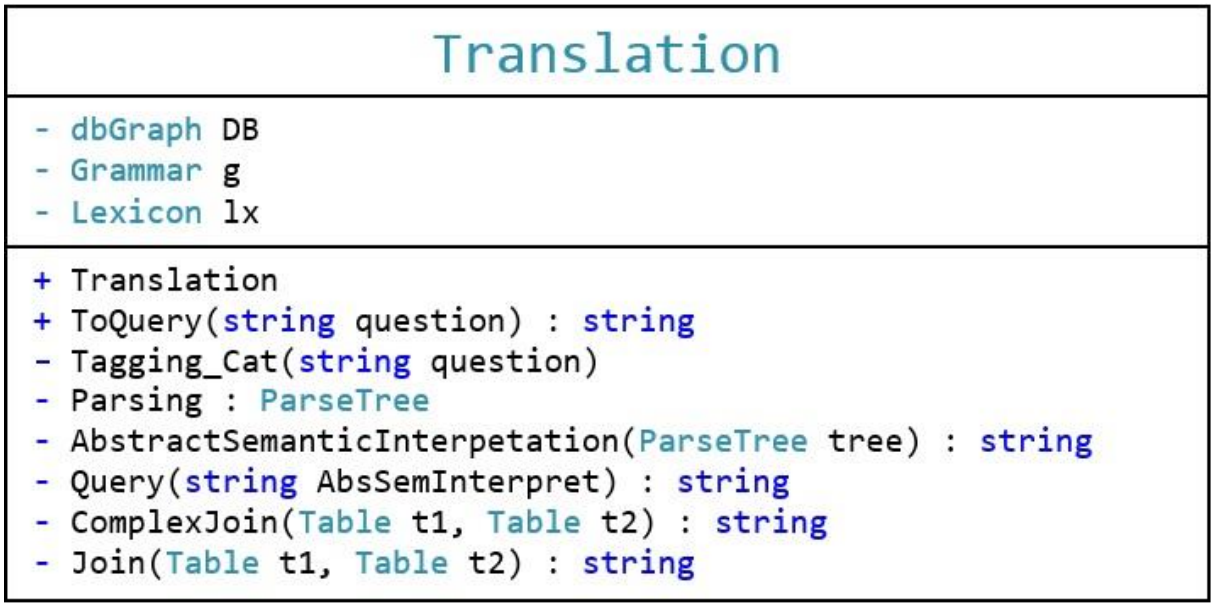

Рис.11: Описание класса Translation

Результат

В результате работы реализовано приложение, в котором в поле ввода вводится вопрос, затем, по нажатии кнопки “Выполнить”, выводится результат запроса. Также есть возможность увидеть сам запрос, нажав “Показать запрос”.

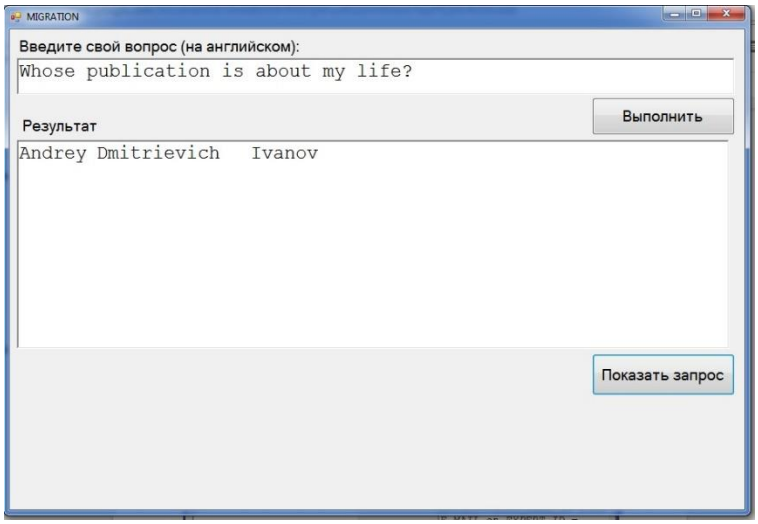

Рис. 12: Интерфейс программы без показа запроса

По нажатии “Спрятать запрос” можно убрать поле с выводом запроса.

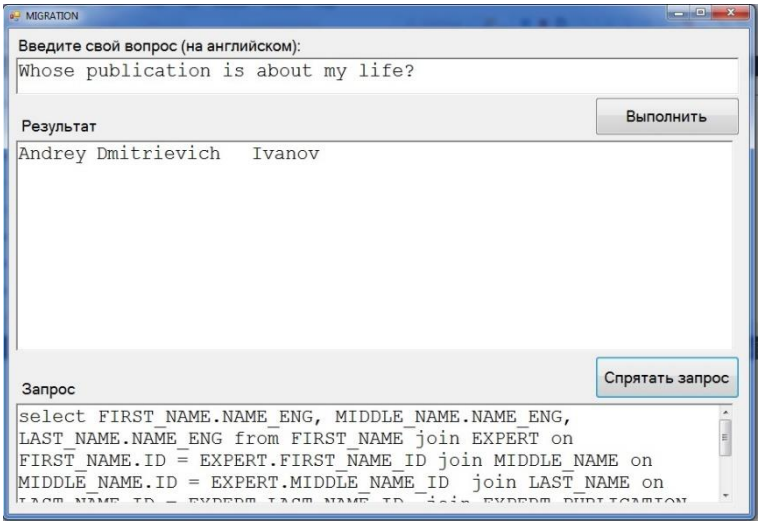

Рис. 13: Интерфейс программы с показом запроса

Проект можно скачать по ссылке https://github.com/shparutask/TranslationSystem. Инструкция по запуску и пользованию приложением находится в Приложении 2.

На данный момент реализовано выполнение типов вопросов, которые указаны в Приложении 1.

Примеры:

1. Whose publication is about my life?

Запрос: *select FIRST_NAME.NAME_ENG, MIDDLE_NAME.NAME_ENG, LAST_NAME.NAME_ENG from FIRST_NAME join EXPERT on FIRST_NAME.ID = EXPERT.FIRST_NAME_ID join MIDDLE_NAME on MIDDLE_NAME.ID = EXPERT.MIDDLE_NAME_ID join LAST_NAME on LAST_NAME.ID = EXPERT.LAST_NAME_ID join EXPERT_PUBLICATION on EXPERT.ID = EXPERT_PUBLICATION.ID_EXPERT join PUBLICATION on PUBLICATION.ID = EXPERT_PUBLICATION.ID_PUBLICATION where PUBLICATION.DESCRIPTION_ENG like '%my life%'*

Ответ: Andrey Dmitrievich Ivanov

2. What is the name of author of publication like my life?

Запрос: *select NAME_ENG from FIRST_NAME join EXPERT on FIRST_NAME.ID = EXPERT.FIRST_NAME_ID join EXPERT_PUBLICATION on EXPERT.ID = EXPERT_PUBLICATION.ID_EXPERT join PUBLICATION on PUBLICATION.ID = EXPERT_PUBLICATION.ID_PUBLICATION where PUBLICATION.DESCRIPTION_ENG like '%my life%'*

Ответ: Andrey

3. What are mails there?

Запрос: *select E_MAIL.MAIL from E_MAIL*

Ответ:

1. eroidsj@sdf.ru
2. kjfsd@df.com
3. sofish718@sdf.ru

Заключение

Итак, на данный момент есть приложение для Windows, настроенное на конкретную базу, которое выдает ответы на вопрос пользователя на английском языке. Реализованы вопросы, указанные в приложении 1.

В дальнейшем планируется реализовать ввод вопросов на русском языке. Но здесь очевидно возникнут трудности из-за особенностей русского языка, таких как склонения существительных, спряжения глаголов и т.п. Очевидно, придется заново писать новую грамматику или корректно прописать синонимы, но это довольно сложная и кропотливая задача.

Используемая литература

1. Alexander Ran, Raimondas Lencevicius. Natural Language Query System for RDF Repositories // Proceedings of the 7-th International Symposium on Natural Language Processing, SLNP. –2007. –6.
2. Florin Brad, Radu Iacob, Ionel Hosu, and Traian Rebedea. Dataset for a Neural Natural Language Interface for Databases (NNLIDB) // Proceedings of the 8-th International Joint Conference on Natural Language Processing. – 2017. – c.906-914.
3. Fred Popowich, Milan Mosny, David Lindberg. Interactive Natural Language Query Construction for Report Generation // Proceedings of the 7-th International Natural Language Generation Conference. –2012. – c.115-119.
4. Nicolas Kuchmann-Beauger. Question Answering System in a Business Intelligence Context // HAL archives-ouvertes.fr. –2017. – c.15-137.
5. Shay Cohen, Toms Bergmanis. A Natural Language Query System in Python/NLTK. – https://github.com/andrrra/Natural-Language-Query-System.

Приложение 1. Типы вопросов, распознаваемые системой

а) Транслируются и выдают верный результат:

What is the ___ of “ ”?

What is the ____ of author/expert who has a publication(s) about/like “ ”?

What is the ____ of author/expert of publication(s) about/like “ ”?

What is the ____ of author/expert of publication(s) with ___ like “ ”?

What is the __ of author/expert of publication(s) which has a ____ like “ ”?

Who has the ___ like “ ”?

What is the ___ of expert/author who has a publication(s) which has a ___ like “ ”?

Whose ___ is “ ”?

What are ____ there?

What are ___ of publications?

What are degrees of experts?

Whose publication is about “ ”?

Which are references of “ ”?

Which are literature of “ ”?

Which are references of publication with ___ like?

Which are literature of publication with ___ like?

What is the ___ of author of publication(s) with used literature like “ ”?

What is the ___ of author of publication which has a used literature like “ ”?

What is the ____ of expert/author who has a publication(s), which has a used literature like “ ”?

Which publication(s) was (were) written by “ ”?

Who has written (a) publication(s) about “ ”?

Who has written (a) publication(s) which has a ___ like “ ”?

б) Выдают неверный ответ или не транслируются:

What is the ___ of author/expert who has a publication(s) about/like “ ” and/or ___ “ ”?

Приложение 2. Инструкция по запуску системы

0. Установить Microsoft SQL Server 2014, SQL Server 2014 Management Studio. Создать и заполнить базу данных в Management Studio. В качестве примера можно запустить Expert_script.sql в Management Studio из папки Scripts (см. Приложение 3).
1. Запустить файл TranslationSystem.exe
2. В поле «Введите название экземпляра SQL Server» ввести имя экземпляра сервера. (Его можно посмотреть, зайдя в Диспетчер конфигурации SQL Server)
3. В поле «Введите название базы данных» ввести название базы данных, с которой планируется работа. (Для примера можно ввести MIGRATION_EXPERT)
4. Нажать Enter, или «Войти» левой кнопкой мыши.
5. Ввести вопрос в поле «Введите свой вопрос».

Приложение 3. Запуск примера скрипта

1. Открыть Management Studio.
2. Верхнем меню выбрать Файл – Открыть – Файл.
3. В диалоговом окне открыть папку Scripts.
4. Выбрать Expert_script.sql.
5. В открывшемся файле выделить первую строку и нажать F5.
6. Внизу появится сообщение об успешно выполненном запросе.
7. Перед первой строкой поставить --. Строка окрасится в зеленый.
8. Нажать F5.
9. База создана. Чтобы заполнить таблицы, в меню базы выбрать нужную таблицу, затем в меню, появившемся по нажатию правой кнопкой мыши, выбрать «Изменить первые 200 строк».

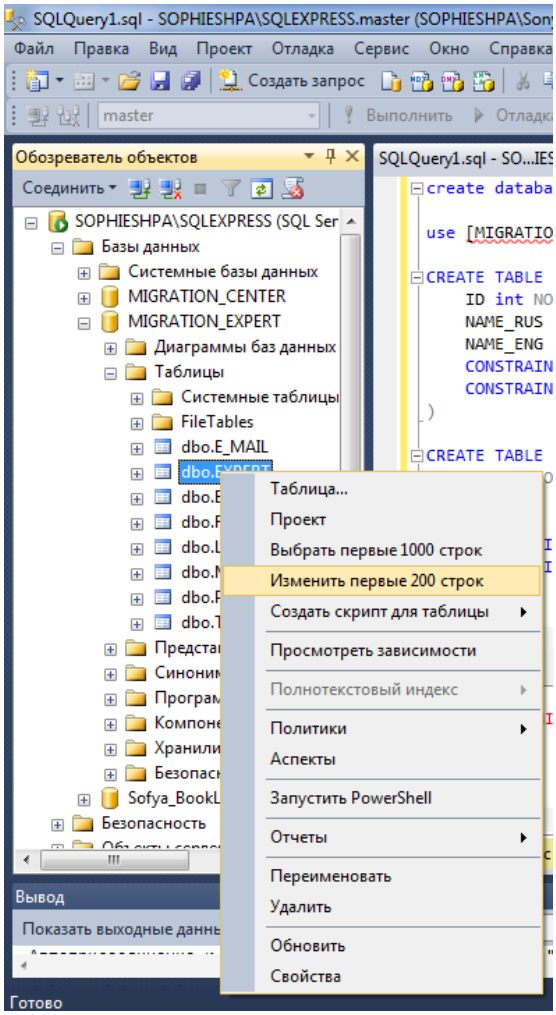

Рис. 14: Интерфейс SQL Server Management Studio – Обозреватель объектов